Reti di Calcolatori T
Prova d'esame
21/12/2020

Cognome: ……
Nome: ……
Matricola: ……

Tempo a disposizione: 3h

È obbligatorio inserire Cognome, Nome, e numero di Matricola all'inizio di ogni file sorgente, pena la non valutazione del compito, che viene stampato in modo automatico solo in caso siano presenti gli elementi detti sopra.
Si devono consegnare **singolarmente tutti i file sorgente e tutti gli eseguibili prodotti** (*per favore, solo quelli relativi ai file sorgente consegnati!!!*).

La prova intende valutare le capacità progettuali e di programmazione sia in **ambiente Java** che in **ambiente C**, pertanto è consigliato sviluppare **entrambe** le soluzioni richieste al meglio.
In entrambi gli esercizi, sia in Java che in C, si effettuino gli opportuni controlli sugli argomenti della richiesta e si gestiscano le eccezioni verso l'utente, tenendo presente i criteri secondo cui si possa ripristinare il funzionamento del programma oppure si debba forzarne la terminazione.
Leggete con attenzione le specifiche del problema prima di impegnarvi "a testa bassa" nello sviluppo delle singole parti. Naturalmente, ci aspettiamo che i componenti da consegnare siano stati provati e siano funzionanti.

****

Si richiede il progetto della gestione dei servizi **OrariTreno**, per la gestione del pannello dei viaggi in arrivo/partenza in una stazione.
I servizi di BusOrari mantengono, per ogni viaggio in partenza o arrivo, le seguenti informazioni: un **identificatore** del viaggio, unico all'interno del sistema; il **tipo** di treno che può essere 'Partenza' o 'Arrivo'; le **città di partenza** e **arrivo**; **l'ora prevista** del viaggio (nella forma **'hh:mm'**); i **minuti di ritardo**; e un **nome di file audio** di annuncio (salvato sul file system locale).
Si vogliono realizzare le funzionalità di gestione:

1. **inserimento di un nuovo viaggio**: questa operazione richiede *l'identificatore del viaggio*, il *tipo*, *l'ora attesa*, e il *nome del file*, ed inserisce il nuovo viaggio nella struttura dati provvedendo anche all'*upload del file*;
2. **visualizzazione di tutti i viaggi di un certo tipo in partenza**: questa operazione richiede il *tipo di viaggio* e restituisce l'elenco di *tutte le corse di quel tipo* (che risultano in partenza o arrivo);
3. **download di tutti i file audio di annuncio per viaggi in arrivo nella prossima ora**: questa operazione richiede *un'ora* (nella forma 'hh:mm') e restituisce scarica dal server al cliente i file audio (come nome del file salvato sul client si usil'identificatore) per i viaggi che arriveranno entro un'ora (si consideri il tempo reale che si ottiene sommando al tempo atteso i minuti di ritardo);
4. **modifica il ritardo di un viaggio**: questa operazione richiede *l'identificatore del viaggio* e il *ritardo*, e aggiorna la struttura dati.

Si progetti con particolare attenzione la **struttura dati** che mantiene lo stato, fino ad un massimo di N viaggi (L, per libero a default o valore equivalente), da implementare opportunamente nei diversi ambienti richiesti, Java e C. Per l'ora si usino strutture più adatte e semplici (es. **due interi**) e si salvino i file nel direttorio corrente.

Identificatore	Tipo	Partenza	Arrivo	Ora attesa	Ritardo	Audio
L	**L**	**L**	**L**	**-1:-1**	**-1**	**L**
SATA1234	**Partenza**	**Bologna**	**Bari**	**12:15**	**0**	**SATA1234.mp4**
MATA3333	**Arrivo**	**Milano**	**Bologna**	**16:30**	**21**	**MATA3333.mp4**
…	**…**			**…**	**…**	**…**
CATA1111	**Partenza**	**Bologna**	**Napoli**	**13:00**	**190**	**CATA1111.mp4**
L	**L**	**L**	**L**	**-1:-1**	**-1**	**-L**

Si considerino e si segnalino le possibilità di interferenze fra le operazioni, evitandole dove necessario.

Parte Java

Sviluppare un'applicazione C/S basata su **socket stream** che realizzi le operazioni remote per:

- *effettuare il download di tutti i file audio di annuncio per viaggi in arrivo nella prossima ora*;
- *inserire un nuovo viaggio.*

Si utilizzi una sola connessione per ogni sessione client. Più in dettaglio:

- Il **cliente** è organizzato come un **processo filtro ciclico che consuma l'input fino a fine file** e, per ogni iterazione del ciclo, chiede all'utente quale tipo di operazione vuole effettuare e realizza le interazioni col server utilizzando **una sola connessione per la intera sessione**; alla ricezione del fine file, libera opportunamente le risorse e termina. Per ogni richiesta ricevuta dall'utente, il client prima invia il tipo di servizio al server, poi gestisce gli invii e le ricezioni necessarie alla realizzazione dello specifico servizio richiesto.
  Nel caso della funzionalità di **download di tutti i file audio di annuncio per viaggi in arrivo nella prossima ora,** il client richiede all'utente l'ora attuale (nella forma 'hh:mm') e la invia al server, quindi riceve i file audio, salvandoli sul file system locale.
  Nel caso della funzionalità di **inserimento di un nuovo viaggio**, il client richiede all'utente e invia al server l'identificatore del viaggio, il tipo, l'ora attesa e nome del file audio; quindi effettua l'upload del file audio, riceve l'esito dell'operazione e lo stampa a video.
- Il **server** è organizzato come un **unico processo che gestisce in modo parallelo** l'interazione coi clienti, generando un figlio per tutta la sessione di richieste da quel client. Per ogni richiesta, il processo figlio che serve la sessione con una prima lettura discrimina il tipo di funzionalità richiesto, poi gestisce opportunamente l'operazione e si pone in attesa di nuove richieste dallo stesso client; alla lettura della fine sessione, il figlio termina.
  Per ogni richiesta di **download di tutti i file audio di annuncio per viaggi in arrivo nella prossima ora**, il figlio riceve l'ora e trasmette al client i file audio relativi ai viaggi di tipo 'Arrivo' che arriveranno nella ora successiva (si consideri il tempo reale degli orari, ottenuto sommando al tempo atteso i minuti di ritardo).
  Per ogni richiesta di **inserimento di un nuovo viaggio**, il figlio riceve l'identificatore del viaggio, il tipo, l'ora attesa e il nome del file audio, riceve il file audio e restituisce al client l'esito dell'operazione, positivo se l'inserimento ha successo, oppure un negativo, ad esempio se un viaggio con lo stesso identificatore è già stato inserito, se la struttura dati è piena o se falliscono alcuni controlli sui dati di input.

Parte C

Utilizzando RPC sviluppare un'applicazione C/S che consenta di effettuare le **operazioni remote** per:

- *visualizzare i viaggi di un certo tipo in partenza (al massimo 6);*
- *modificare il ritardo di un viaggio.*

Il progetto prevede che il server metta a disposizione due procedure (parte di una interfaccia specificata in XDR e contenuta nel file *RPC_xFile.x*) invocabili in remoto dal client:

- La procedura **visualizza_lista** accetta come parametro d'ingresso una stringa che rappresenta il tipo; quindi il server restituisce (se ci sono) la lista **delle prime N corse** (N<=6) di quel tipo che risultano in partenza.
- La procedura **modificare_ritardo_viaggio** accetta come parametri d'ingresso una struttura dati contenente l'identificatore del viaggio e il ritardo; quindi il server aggiorna la struttura restituendo l'esito dell'operazione, 0 se l'aggiornamento è andato a buon fine, -1 altrimenti, ad esempio, se il viaggio non è presente o uno dei controlli sugli input fallisce.

Si progettino inoltre i sorgenti:

- ***RPC_Server*** (contenuta nel file *RPC_Server.c*), che implementa le procedure del server invocabili in remoto;
- ***RPC_Client*** (contenuta nel file *RPC_Client.c*), il processo filtro che realizza l'interazione con l'utente, **propone ciclicamente i servizi** che utilizzano le due procedure remote, e stampa a video i risultati, fino alla fine dello stream di input.